

Filename: IEHooker.idl

Page 1 of 23

```

// IEHooker.idl : IDL source for IEHooker.dll
//

// This file will be processed by the MIDL tool to
// produce the type library (IEHooker.tlb) and marshalling code.

import "oaidl.idl";
import "ocidl.idl";

[
    object,
    uuid(16122F01-9713-11D3-9744-005004116944),
    dual,
    helpstring("ICIEHooker Interface"),
    pointer_default(unique)
]
interface ICIEHooker : IDispatch
{
};

[
    uuid(16122EF1-9713-11D3-9744-005004116944),
    version(1.0),
    helpstring("IEHooker 1.0 Type Library")
]
library IEHOOKERLib
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

    [
        uuid(16122F03-9713-11D3-9744-005004116944),
        helpstring("_ICIEHookerEvents Interface")
    ]
    dispinterface _ICIEHookerEvents
    {
        properties:
        methods:
        [id(1), helpstring("method MakeCall")] HRESULT MakeCall(BSTR
bstr);
    };

    [
        uuid(16122F02-9713-11D3-9744-005004116944),
        helpstring("CIEHooker Class")
    ]
    coclass CIEHooker
    {
        [default] interface ICIEHooker;
        [default, source] dispinterface _ICIEHookerEvents;
    };
};

```

```
#ifndef CHECK_NUM_H
#define CHECK_NUM_H

#include "mshtml.h"

#define DO_IDLE 0
#define DO_START 1
#define DO_CONTINUE 2
#define DO_SUCCESSEND 3
#define DO_FAILEND 4
typedef struct DATA_tag
{
    char* input_buf;
    int input_size;
    char* output_buf;
    int output_size;
    int iCheckStatus;
    BOOL bUserStop;

    struct DATA_tag* pNext;
    IHTMLDocument2* pDocument;
}STRUCT_PROCESS;

void checkPhoneNumThread(void *pVoid);
void checkPhoneNumProc(STRUCT_PROCESS *pData);
BOOL checkPhoneNum(STRUCT_PROCESS *pData);

#endif
```

11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125. 126. 127. 128. 129. 130. 131. 132. 133. 134. 135. 136. 137. 138. 139. 140. 141. 142. 143. 144. 145. 146. 147. 148. 149. 150. 151. 152. 153. 154. 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179. 180. 181. 182. 183. 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195. 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207. 208. 209. 210. 211. 212. 213. 214. 215. 216. 217. 218. 219. 220. 221. 222. 223. 224. 225. 226. 227. 228. 229. 230. 231. 232. 233. 234. 235. 236. 237. 238. 239. 240. 241. 242. 243. 244. 245. 246. 247. 248. 249. 250. 251. 252. 253. 254. 255. 256. 257. 258. 259. 260. 261. 262. 263. 264. 265. 266. 267. 268. 269. 270. 271. 272. 273. 274. 275. 276. 277. 278. 279. 280. 281. 282. 283. 284. 285. 286. 287. 288. 289. 290. 291. 292. 293. 294. 295. 296. 297. 298. 299. 300. 301. 302. 303. 304. 305. 306. 307. 308. 309. 310. 311. 312. 313. 314. 315. 316. 317. 318. 319. 320. 321. 322. 323. 324. 325. 326. 327. 328. 329. 330. 331. 332. 333. 334. 335. 336. 337. 338. 339. 340. 341. 342. 343. 344. 345. 346. 347. 348. 349. 350. 351. 352. 353. 354. 355. 356. 357. 358. 359. 360. 361. 362. 363. 364. 365. 366. 367. 368. 369. 370. 371. 372. 373. 374. 375. 376. 377. 378. 379. 380. 381. 382. 383. 384. 385. 386. 387. 388. 389. 390. 391. 392. 393. 394. 395. 396. 397. 398. 399. 400. 401. 402. 403. 404. 405. 406. 407. 408. 409. 410. 411. 412. 413. 414. 415. 416. 417. 418. 419. 420. 421. 422. 423. 424. 425. 426. 427. 428. 429. 430. 431. 432. 433. 434. 435. 436. 437. 438. 439. 440. 441. 442. 443. 444. 445. 446. 447. 448. 449. 450. 451. 452. 453. 454. 455. 456. 457. 458. 459. 460. 461. 462. 463. 464. 465. 466. 467. 468. 469. 470. 471. 472. 473. 474. 475. 476. 477. 478. 479. 480. 481. 482. 483. 484. 485. 486. 487. 488. 489. 490. 491. 492. 493. 494. 495. 496. 497. 498. 499. 500. 501. 502. 503. 504. 505. 506. 507. 508. 509. 510. 511. 512. 513. 514. 515. 516. 517. 518. 519. 520. 521. 522. 523. 524. 525. 526. 527. 528. 529. 530. 531. 532. 533. 534. 535. 536. 537. 538. 539. 540. 541. 542. 543. 544. 545. 546. 547. 548. 549. 550. 551. 552. 553. 554. 555. 556. 557. 558. 559. 560. 561. 562. 563. 564. 565. 566. 567. 568. 569. 570. 571. 572. 573. 574. 575. 576. 577. 578. 579. 580. 581. 582. 583. 584. 585. 586. 587. 588. 589. 590. 591. 592. 593. 594. 595. 596. 597. 598. 599. 600. 601. 602. 603. 604. 605. 606. 607. 608. 609. 610. 611. 612. 613. 614. 615. 616. 617. 618. 619. 620. 621. 622. 623. 624. 625. 626. 627. 628. 629. 630. 631. 632. 633. 634. 635. 636. 637. 638. 639. 640. 641. 642. 643. 644. 645. 646. 647. 648. 649. 650. 651. 652. 653. 654. 655. 656. 657. 658. 659. 660. 661. 662. 663. 664. 665. 666. 667. 668. 669. 670. 671. 672. 673. 674. 675. 676. 677. 678. 679. 680. 681. 682. 683. 684. 685. 686. 687. 688. 689. 690. 691. 692. 693. 694. 695. 696. 697. 698. 699. 700. 701. 702. 703. 704. 705. 706. 707. 708. 709. 710. 711. 712. 713. 714. 715. 716. 717. 718. 719. 720. 721. 722. 723. 724. 725. 726. 727. 728. 729. 730. 731. 732. 733. 734. 735. 736. 737. 738. 739. 740. 741. 742. 743. 744. 745. 746. 747. 748. 749. 750. 751. 752. 753. 754. 755. 756. 757. 758. 759. 760. 761. 762. 763. 764. 765. 766. 767. 768. 769. 770. 771. 772. 773. 774. 775. 776. 777. 778. 779. 780. 781. 782. 783. 784. 785. 786. 787. 788. 789. 790. 791. 792. 793. 794. 795. 796. 797. 798. 799. 800. 801. 802. 803. 804. 805. 806. 807. 808. 809. 810. 811. 812. 813. 814. 815. 816. 817. 818. 819. 820. 821. 822. 823. 824. 825. 826. 827. 828. 829. 830. 831. 832. 833. 834. 835. 836. 837. 838. 839. 840. 841. 842. 843. 844. 845. 846.

```

{
    for(int i = 0 ; i < (int)strlen(num) ; i++)
    {
        if( (num[i] >= '0' && num[i] <= '9') )
        {
            nDigitCount++;
            num[i] = '#';
        }
        else if ( (num[i] >= 'A' && num[i] <= 'Z') )
        {
            num[i] = '#';
        }
    }
    for(i = (int)strlen(num) - 1 ; i >= 0 ; i--)
    {
        if(num[i] == '#') break;
        if( num[i] == ' ' ) num[i] = NULL;
    }
    for(i = 0 ; i < MAX_MAP_COUNT ; i++)
    {
        if( (strcmp(szNumMapTable[i],num) == 0)
            && nDigitCount >= 3 )
        {
            return TRUE;
        }
    }
    return FALSE;
}

BOOL isLinkTag( char* strTag)
{
    if(strcmp(strTag,"a") == 0 || strcmp(strTag,"A") == 0)
        return TRUE;
    return FALSE;
}

BOOL isLinkEndTag( char* strTag)
{
    if(strcmp(strTag,"/a") == 0 || strcmp(strTag,"/A") == 0)
        return TRUE;
    return FALSE;
}

BOOL checkPhoneNum(STRUCT_PROCESS *pData)
{
    char *copyText;
    char ch;
    BOOL bRtn = FALSE;
    BOOL bSkip = FALSE;
    BOOL bNum = FALSE;

    BOOL bLinkTag = FALSE;
    BOOL bTag = FALSE;

```

```

int checkSpace = nInsertLen * 10; //strlen(szInsertString1) * 10;

char phoneNum[MAX_PATH];
char dummyPhoneNum[MAX_PATH];
char strTag[MAX_PATH];

if(pData->iCheckStatus != DO_START) return FALSE;

copyText = new char[pData->input_size + checkSpace];
pData->output_buf = new char[pData->input_size + checkSpace];

int insize = pData->output_size = pData->input_size;

memset(phoneNum, NULL, MAX_PATH);
memset(dummyPhoneNum, NULL, MAX_PATH);
memset(copyText, NULL, insize);
memset(strTag, NULL, MAX_PATH);

int j = 0;
int k = 0;
int l = 0;
int m = 0;

pData->iCheckStatus = DO_CONTINUE;

for(int i = 0 ; i < insize ; i++) {
    if(pData->bUserStop) {
        bRtn = FALSE;
        break;
    }

    ch = pData->input_buf[i];

    if ( ( ch >= '0' && ch <= '9' )
        || ( ch >= 'A' && ch <= 'Z' && TRUE == bNum ) )
    {
        if( bSkip == FALSE ) {
            phoneNum[j++] = ch;
            dummyPhoneNum[l++] = ch;
            bNum = TRUE;
        } else {
            copyText[k++] = ch;
        }
    }
    else
    {
        switch(ch) {
            case '.':
                if( bSkip == FALSE ) {
                    if(bNum && (l == 3 || l == 7)) {
                        dummyPhoneNum[l++] = ch;
                    }
                    // the '.' might be a sentence terminator
                    else if (bNum && (l == 8 || l == 12 || l == 14))
                    {
                        goto check_routine;
                    }
                }
            }
        }
    }
}

```

```

        else {
            for(int i = 0 ; i <
(int)strlen(dummyPhoneNum) ; i++) {
                copyText[k++] = dummyPhoneNum[i];
            }
            copyText[k++] = ch;
            bNum = FALSE;
            memset (dummyPhoneNum, NULL, MAX_PATH);
            memset (phoneNum, NULL, MAX_PATH);
            j = 0;
            l = 0;
        }
    } else {
        copyText[k++] = ch;
    }
    break;
case '/':
    if( bSkip == FALSE ) {
        if(bNum && (l == 3 || l == 7)) {
            dummyPhoneNum[l++] = ch;
        } else {
            for(int i = 0 ; i <
(int)strlen(dummyPhoneNum) ; i++) {
                copyText[k++] = dummyPhoneNum[i];
            }
            copyText[k++] = ch;
            bNum = FALSE;
            memset (dummyPhoneNum, NULL, MAX_PATH);
            memset (phoneNum, NULL, MAX_PATH);
            j = 0;
            l = 0;
        }
    } else {
        copyText[k++] = ch;
    }
    break;
case ')':
    if( bSkip == FALSE ) {
        if(bNum) {
            if(l == 4 || l == 7 || l == 13)
                dummyPhoneNum[l++] = ch;
            else
                goto check_routine;
        } else {
            for(int i = 0 ; i <
(int)strlen(dummyPhoneNum) ; i++) {
                copyText[k++] = dummyPhoneNum[i];
            }
            copyText[k++] = ch;
            bNum = FALSE;
            memset (dummyPhoneNum, NULL, MAX_PATH);
            memset (phoneNum, NULL, MAX_PATH);
            j = 0;
            l = 0;
        }
    }
} else {

```

[illegible]

```

        default:
        check_routine:
            if( bSkip == FALSE ) {
                if(bNum) {
                    if( bLinkTag == TRUE) {
                        for(int i = 0 ; i <
(int)strlen(dummyPhoneNum) ; i++) {
                            copyText[k++] =
dummyPhoneNum[i];
                        }
                        memset(dummyPhoneNum, NULL, MAX_PATH);
                        l = 0;
                        copyText[k++] = ch;
                    } else {
                        if(!IsValidNum(dummyPhoneNum)) {
                            for(int i = 0 ; i <
(int)strlen(dummyPhoneNum) ; i++) {
                                copyText[k++] =
dummyPhoneNum[i];
                            }

                            memset(dummyPhoneNum, NULL, MAX_PATH);
                            l = 0;
                            copyText[k++] = ch;
                        } else {
                            for(int i = 0; i <
(int)strlen(szInsertString1) ; i++) {
                                copyText[k++] =
szInsertString1[i];
                            }
                            for(i = 0 ; i <
(int)strlen(dummyPhoneNum) ; i++) {
                                if(dummyPhoneNum[i] >= '0'
&& dummyPhoneNum[i] <= '9')
                                    copyText[k++] =
dummyPhoneNum[i];
                            }
                            for(i = 0; i <
(int)strlen(szInsertString2) ; i++) {
                                copyText[k++] =
szInsertString2[i];
                            }
                            for(i = 0 ; i <
(int)strlen(dummyPhoneNum) ; i++) {
                                copyText[k++] =
dummyPhoneNum[i];
                            }

                            memset(dummyPhoneNum, NULL, MAX_PATH);
                            l = 0;

                            for(i = 0; i <
(int)strlen(szInsertString3) ; i++) {
                                copyText[k++] =
szInsertString3[i];
                            }
                            copyText[k++] = ch;

```



```

        bRtn = TRUE;
    }
}
memset(phoneNum, NULL, MAX_PATH);
bNum = FALSE;
j = 0;
}
else
{
    // bNum is false
    bLinkTag = FALSE;
    copyText[k++] = ch;
}

if(ch == '<') {
    memset(strTag, NULL, MAX_PATH);
    int idx = 1;
    while (1) {
        if((insize >= i + idx) &&
            (pData->input_buf[i + idx] ==
'>' || pData->input_buf[i + idx] == ' '))
        {
            if(isLinkTag(strTag))
                bLinkTag = TRUE;
            else if(isLinkEndTag(strTag))
                bLinkTag = FALSE;
            break;
        } else {
            if(insize >= i + idx)
                strTag[idx-1] = pData-
>input_buf[i + idx];

            else
                break;
        }
        idx++;
    }
    bSkip = TRUE;
}
} else { // bSkip is true
    copyText[k++] = ch;

    // reset bLinkTag if reached end anchor
    if(ch == '<') {
        memset(strTag, NULL, MAX_PATH);
        int idx = 1;
        while (1) {
            if((insize >= i + idx) &&
                (pData->input_buf[i + idx] ==
'>' || pData->input_buf[i + idx] == ' '))
            {
                if(isLinkTag(strTag))
                    bLinkTag = TRUE;
                else if(isLinkEndTag(strTag))
                    bLinkTag = FALSE;
                break;
            } else {
                if(insize >= i + idx)

```

[illegible]


```
// CIEHooker.h : Declaration of the CCIEHooker
```

```
#ifndef __CIEHOOKER_H_
#define __CIEHOOKER_H_
```

```
#include "resource.h"          // main symbols
#include "IEHookerCP.h"
#include "ExDispID.h"
#include "mshtmdid.h"
#include "mshtml.h"
#include <strstrea.h>
```

```
#include "checkNum.h"
```

```
////////////////////////////////////
// CCIEHooker
```

```
class ATL_NO_VTABLE CCIEHooker :
public CComObjectRootEx<CComSingleThreadModel>,
public CComCoClass<CCIEHooker, &CLSID_CIEHooker>,
public IObjectWithSiteImpl<CCIEHooker>,
public ISupportErrorInfo,
public IConnectionPointContainerImpl<CCIEHooker>,
public IDispatchImpl<ICIEHooker, &IID_ICIEHooker, &LIBID_IEHOOKERLib>,
public CProxy_ICIEHookerEvents< CCIEHooker >
```

```
{
public:
```

```
DECLARE_REGISTRY_RESOURCEID(IDR_CIEHOOKER)
```

```
DECLARE_PROTECT_FINAL_CONSTRUCT()
```

```
BEGIN_COM_MAP(CCIEHooker)
    COM_INTERFACE_ENTRY(ICIEHooker)
    COM_INTERFACE_ENTRY(IDispatch)
    COM_INTERFACE_ENTRY(ISupportErrorInfo)
    COM_INTERFACE_ENTRY(IConnectionPointContainer)
    COM_INTERFACE_ENTRY(IObjectWithSite)
    COM_INTERFACE_ENTRY_IMPL(IConnectionPointContainer)
```

```
END_COM_MAP()
BEGIN_CONNECTION_POINT_MAP(CCIEHooker)
    CONNECTION_POINT_ENTRY(DIID__ICIEHookerEvents)
END_CONNECTION_POINT_MAP()
```

```
public:
```

```
    CCIEHooker();
    ~CCIEHooker();
```

```
    //
    // ISupportsErrorInfo
    //
    STDMETHODCALLTYPE (InterfaceSupportsErrorInfo) (REFIID riid);
```

```
    //
    // IDispatch Methods
    //
    STDMETHODCALLTYPE (Invoke) (DISPID dispidMember, REFIID riid, LCID lcid, WORD wFlags,
        DISPPARAMS * pdispparams, VARIANT * pvarResult,
```

20230424 14:29:30

[illegible]

```

// CIEHooker.cpp : Implementation of CCIEHooker
#include "stdafx.h"
#include "IEHooker.h"
#include "CIEHooker.h"
#include <comdef.h>

////////////////////////////////////
// CCIEHooker
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

STRUCT_PROCESS *g_pHeaderData = NULL;
DWORD iCurrentTick = 0;

void EnablePhoneParse(BOOL bEnablePhoneParse)
{
    HKEY hAppKey;
    DWORD dwDisposition;

    if(RegCreateKeyEx(HKEY_LOCAL_MACHINE, TEXT("Software\\DialPad.com"), 0,
NULL,
        REG_OPTION_NON_VOLATILE, KEY_WRITE, NULL, &hAppKey, &dwDisposition)
        == ERROR_SUCCESS)
    {
        RegSetValueEx(hAppKey, TEXT("EnablePhoneParse"), 0,
REG_BINARY, (LPBYTE)&bEnablePhoneParse, sizeof(WORD));
        RegCloseKey(hAppKey);
    }
}

BOOL IsEnablePhoneParse()
{
    HKEY hAppKey;
    DWORD bEnablePhoneParse = 0;
    DWORD ulSize;

    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE, TEXT("Software\\DialPad.com"), 0,
KEY_READ,
        &hAppKey) == ERROR_SUCCESS)
    {
        ulSize = sizeof(DWORD);
        RegQueryValueEx(hAppKey, TEXT("EnablePhoneParse"), NULL,
NULL, (LPBYTE)&bEnablePhoneParse, &ulSize);
        RegCloseKey(hAppKey);
    }
    return (BOOL)bEnablePhoneParse;
}

CCIEHooker::CCIEHooker()
{
}

CCIEHooker::~~CCIEHooker()
{
}

```

```

}

STDMETHODIMP CCIEHooker::InterfaceSupportsErrorInfo(REFIID riid)
{
    /*    static const IID* arr[] =
        {
            &IID_ICIEHooker
        };
        for (int i=0; i < sizeof(arr) / sizeof(arr[0]); i++)
        {
            if (InlineIsEqualGUID(*arr[i],riid))
                return S_OK;
        }
    */    return S_FALSE;
}

BOOL CCIEHooker::ManageConnection(enum ConnectType eConnectType)
{
    HRESULT hr;

    if (!m_spWebBrowser2) return S_OK;

    CComQIPtr<IConnectionPointContainer>
        &IID_IConnectionPointContainer> spCPCContainer(m_spWebBrowser2);

    if (spCPCContainer != NULL) {
        CComPtr<IConnectionPoint> spConnectionPoint;
        hr = spCPCContainer->FindConnectionPoint(DIID_DWebBrowserEvents2,
        &spConnectionPoint);
        if (SUCCEEDED(hr)) {
            if (eConnectType == Advise) {
                hr = spConnectionPoint->Advise((IDispatch*)this, &m_dwCookie);
            } else {
                hr = spConnectionPoint->Unadvise(m_dwCookie);
            }
        }
    }
    return (SUCCEEDED(hr));
}

STDMETHODIMP CCIEHooker::SetSite(IUnknown *pUnkSite)
{
    USES_CONVERSION;

    if (!pUnkSite) {
        ATLTRACE("\nSetSite(): pUnkSite is NULL\n\n");
    } else {
        m_spWebBrowser2 = pUnkSite;
        if (m_spWebBrowser2) {
            if (!ManageConnection(Advise)) {
                ATLTRACE("Failure sinking events from IWebBrowser2");
            }
        }
    }
    return S_OK;
}

```

[illegible]


```

        default:
            break;
    }
    return S_OK;
}

void CCIEHooker::checkAllData()
{
    STRUCT_PROCESS* pCurData = NULL;
    STRUCT_PROCESS* pNextData = NULL;

    if(g_pHeaderData) {
        try {
            pCurData = pNextData = g_pHeaderData;
            while(pNextData) {
                pCurData = pNextData;
                if(pCurData->iCheckStatus == DO_SUCCESSEND)
                    checkUpdate(pCurData);
                pNextData = pCurData->pNext;
            }
        } catch(...) {
            ATLTRACE(_T("Unspecified exception thrown in
checkAllData\n"));
        }
    }
}

void CCIEHooker::checkUpdate(STRUCT_PROCESS* pData)
{
    if(pData) {
        if(pData->iCheckStatus == DO_SUCCESSEND) {
            try {
                pData->iCheckStatus = DO_IDLE;
                int size = MultiByteToWideChar(CP_ACP, 0, pData->
                >output_buf, -1, 0, 0);
                if(pData->output_size - pData->input_size >= 0) {
                    IHTMLDocument* pbody = NULL;
                    IHTMLDocument2* pHtmlDocument = pData->pDocument;
                    if (pHtmlDocument) {
                        if(SUCCEEDED(pHtmlDocument->get_body( &pbody
                    ))) {
                            if(pbody) {
                                BSTR bstr;
                                OLECHAR* olestr = new
                                OLECHAR[size+1];

                                olestr[0] = 0;
                                int irtn =
                                MultiByteToWideChar(CP_ACP, 0, pData->output_buf, size, olestr, size);
                                bstr = SysAllocString(olestr);
                                pbody->put_innerHTML(bstr);
                                SysFreeString(bstr);
                                delete[] olestr;
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        } catch(_com_error Error) {
            ATLTRACE(Error.ErrorMessage());
        } catch(...) {
            ATLTRACE(_T("Unspecified exception thrown in
checkUpdate\n"));
        }
    }
}

BOOL checkChange(IHTMLDocument2* pDocument)
{
    STRUCT_PROCESS *pNextData = NULL;
    STRUCT_PROCESS *pCurData = NULL;

    if(g_pHeaderData == NULL)
        return TRUE;
    if(pDocument != g_pHeaderData->pDocument)
        return TRUE;
    pCurData = pNextData = g_pHeaderData->pNext;

    try {
        IHTMLFramesCollection2* pFrameset = NULL;
        pDocument->get_frames(&pFrameset);
        if(pFrameset) {
            IHTMLWindow2* pWindow2Next;
            IHTMLDocument2* pNextDoc;
            IHTMLElement* pbody;
            long len;
            pFrameset->get_length(&len);
            for(long i = 0; i < len ; i++) {
                _variant_t va(i, VT_I4);
                VARIANT _result;
                VariantInit(&_result);

                pFrameset->item(&va, &_result);

                pWindow2Next = (IHTMLWindow2*)_result.pdispVal;
                pWindow2Next->get_document(&pNextDoc);

                pCurData = pNextData;
                if(pCurData) {
                    if(pCurData->pDocument != pNextDoc) return TRUE;

                    pNextData = pCurData->pNext;
                    if(pNextDoc) {
                        if(checkChange(pNextDoc)) return TRUE;
                    }
                } else {
                    if(pNextDoc) {
                        if(SUCCEEDED(pNextDoc->get_body(&pbody))) {
                            if(pbody) {
                                BSTR bstr;
                                if(SUCCEEDED(pbody->
>get_innerHTML(&bstr))) {
                                    if (bstr) return TRUE;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
    if(checkChange(pNextDoc)) return TRUE;
}
}
}
} catch(...) {
    ATLTRACE(_T("Unspecified exception checkChange\n"));
}
return FALSE;
}

void CCIEHooker::EnumFrames()
{
    IDispatch * pDisp;
    IHTMLDocument2* pDocument;
    try {
        if(SUCCEEDED(m_spWebBrowser2->get_Document(&pDisp))) {
            if(SUCCEEDED(pDisp-
>QueryInterface(IID_IHTMLDocument2, (void**)&pDocument))) {
                if(pDocument) {
                    BOOL bcheck = checkChange(pDocument);
                    if(bcheck) {
                        RemoveAll();
                        RecurseWindows(pDocument);
                    }
                } else {
                    RemoveAll();
                }
            }
        } catch(...) {
            ATLTRACE(_T("Unspecified exception EnumFrames\n"));
        }
    }
}

void CCIEHooker::RecurseWindows(IHTMLDocument2* pDocument)
{
    IHTMLElement* pbody;

    if(pDocument == NULL) return;

    try {
        pDocument->get_body( &pbody );
        if(pbody) {
            GetPageBody(pDocument);

            // Get the IDispatch of the document
            LPDISPATCH lpDisp = pDocument;
            //lpDisp = m_webBrowser.GetDocument();

            if (lpDisp)
            {
                IOleContainer* pContainer;

                // Get the container

```

CCIEHooker.cpp

```

        HRESULT hr = lpDisp->QueryInterface(IID_IoleContainer,
(void**)&pContainer);
        lpDisp->Release();

        if (FAILED(hr))
        {
            return;
        }

        IEnumUnknown* pEnumerator;

        hr = pContainer->EnumObjects(OLECONTF_EMBEDDINGS,
&pEnumerator);
        pContainer->Release();

        if (FAILED(hr))
        {
            return;
        }

        IUnknown* pUnk;
        ULONG uFetched;

        // Enumerate all the frames and process their html info
        for (UINT i = 0; S_OK == pEnumerator->Next(1, &pUnk,
&uFetched); i++)
        {
            IWebBrowser2* pBrowser;

            hr = pUnk->QueryInterface(IID_IWebBrowser2,
(void**)&pBrowser);
            pUnk->Release();

            if (SUCCEEDED(hr))
            {
                IDispatch * pDisp;
                IHTMLDocument2* pDocument;
                try {
                    if(SUCCEEDED(pBrowser-
>get_Document(&pDisp))) {
                        if(SUCCEEDED(pDisp-
>QueryInterface(IID_IHTMLDocument2, (void**)&pDocument))) {
                            if(pDocument) {

                                RecurseWindows(pDocument);

                                pDocument-
>Release();
                            }
                        }
                    } catch(...) {
                        ATLTRACE(_T("Unspecified exception
EnumFrames\n"));
                    }
                    pBrowser->Release();
                }
            }
        }
    }

```

```

        }
        pEnumerator->Release();
    }
}
} catch(...) {
    ATLTRACE(_T("Unspecified exception RecurseWindows\n"));
}
}

BOOL CCIEHooker::GetPageBody(IHTMLDocument2* pDocument)
{
    BSTR bstr;
    IHTMLElement* pbody = NULL;
    BOOL bSuccess = FALSE;
    try {
        STRUCT_PROCESS* pData = new STRUCT_PROCESS;
        memset(pData, NULL, sizeof(STRUCT_PROCESS));

        if (SUCCEEDED(pDocument->get_body(&pbody))) {
            if(pbody) {
                pbody->get_innerHTML(&bstr);
                if (bstr) {
                    int bytes=WideCharToMultiByte(CP_ACP,0,bstr,-
1,0,0,0,0);

                    if (bytes) {
                        char *buf=new char[bytes];
                        memset(buf,NULL,bytes);
                        WideCharToMultiByte(CP_ACP,0,bstr,-
1,buf,bytes,0,0);

                        pData->input_buf = buf;
                        pData->input_size = bytes;
                        pData->pDocument = pDocument;
                        checkPhoneNumProc(pData);

                        bSuccess = TRUE;
                    }
                }
            }
        }
        if(bSuccess) AddData(pData);
        else RemoveData(pData);
    } catch(...) {
        ATLTRACE(_T("Unspecified exception thrown in GetPageBody\n"));
    }
    return TRUE;
}

BOOL CCIEHooker::AddData(STRUCT_PROCESS* pData)
{
    STRUCT_PROCESS* pCurData;
    STRUCT_PROCESS* pNextData;

    try {
        if(g_pHeaderData) {
            pCurData = pNextData = g_pHeaderData;
            while(pNextData) {

```

```

        pCurData = pNextData;
        pNextData = pNextData->pNext;
    }
    pCurData->pNext = pData;
} else {
    g_pHeaderData = pData;
}
} catch(...) {
    ATLTRACE(_T("Unspecified exception thrown in AddData\n"));
}
return TRUE;
}

BOOL CCIEHooker::RemoveAll()
{
    try {
        while(g_pHeaderData) {
            STRUCT_PROCESS* pCurData = g_pHeaderData;
            g_pHeaderData = pCurData->pNext;
            RemoveData(pCurData);
        }
        g_pHeaderData = NULL;
    } catch(...) {
        ATLTRACE(_T("Unspecified exception thrown in RemoveAll\n"));
    }
    return TRUE;
}

BOOL CCIEHooker::RemoveData(STRUCT_PROCESS* pData)
{
    try {
        if(pData != NULL) {
            pData->bUserStop = TRUE;
            while(pData->iCheckStatus == DO_CONTINUE) {
                Sleep(10);
            }

            if(pData->pDocument) {
                pData->pDocument->Release();
                pData->pDocument = NULL;
            }
            if(pData->input_buf) {
                delete[] pData->input_buf;
                pData->input_buf = NULL;
            }
            if(pData->output_buf) {
                delete[] pData->output_buf;
                pData->output_buf = NULL;
            }
            delete pData;
            pData = NULL;
        }
    } catch(...) {
        ATLTRACE(_T("Unspecified exception thrown in RemoveData\n"));
    }
    return TRUE;
}

```

20220724 12:39:36

HKCR

```
{
    IEHooker.CIEHooker.1 = s 'CIEHooker Class'
    {
        CLSID = s '{16122F02-9713-11D3-9744-005004116944}'
    }
    IEHooker.CIEHooker = s 'CIEHooker Class'
    {
        CLSID = s '{16122F02-9713-11D3-9744-005004116944}'
        CurVer = s 'IEHooker.CIEHooker.1'
    }
    NoRemove CLSID
    {
        ForceRemove {16122F02-9713-11D3-9744-005004116944} = s 'CIEHooker
Class'
```

```
    {
        ProgID = s 'IEHooker.CIEHooker.1'
        VersionIndependentProgID = s 'IEHooker.CIEHooker'
        ForceRemove 'Programmable'
        InprocServer32 = s '%MODULE%'
        {
            val ThreadingModel = s 'Apartment'
        }
        'TypeLib' = s '{16122EF1-9713-11D3-9744-005004116944}'
    }
}
```

HKLM

```
{
    SOFTWARE
    {
        Microsoft
        {
            Windows
            {
                CurrentVersion
                {
                    Explorer
                    {
                        'Browser Helper Objects'
                        {
                            {16122F02-9713-11D3-9744-005004116944}
                        }
                    }
                }
            }
        }
    }
}
```